Here is the perfect code cryptosystem, played out by Alice and Bob:

(1) Bob creates a graph G (his public key) that contains a perfect code P (his private key). He keeps P to himself and shares G with everyone.

(2) Alice wants to send an integer N to Bob. To do this, she chooses one integer number n_i for each vertex v_i in Bob's graph G, the only restriction on the n_i's being $n_1 + n_2 + \dots + n_k = N$. (They sum to N)

(3) Having labelled all the vertices of Bob's graph G with numbers n_i, Alice hides her message as follows:

For each vertex v, replace the label of v as follows: If v is adjacent to $v_1, v_2, \dots, v_m$, the new label of v is $n_v + n_1 + n_2 + \dots + n_m$. (I.e: add up the labels of the vertices adjacent to v and v's label). (Clumping) She sends the newly labeled graph to Bob.

(4) Bob adds up the labels of the vertices in P. The result is N.

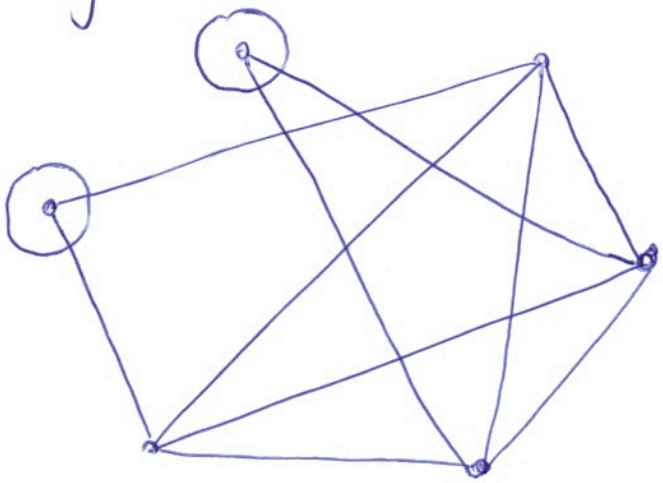
Why does Bob get N?

(upon adding)

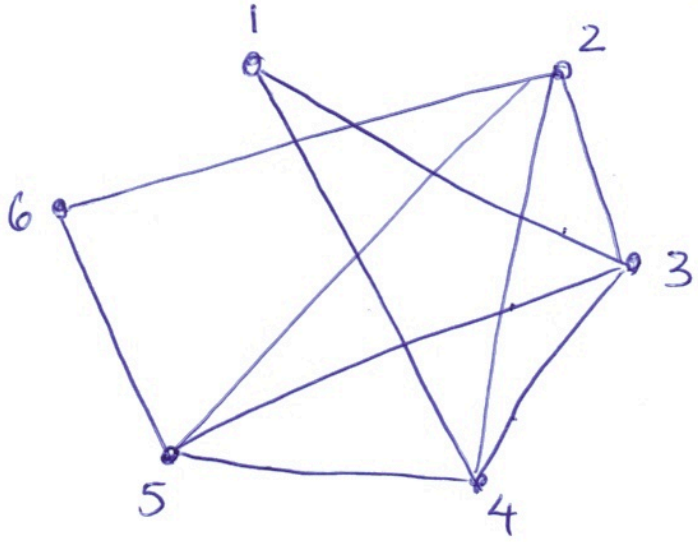
Every vertex in G is connected to exactly one vertex in P. So when you sum the clumped labels of P, you are exactly summing all of the original labels of all vertices in G!

Example: We will do a graph with only a few vertices, which of course is not secure. Bob would do better to make his graph more complicated.

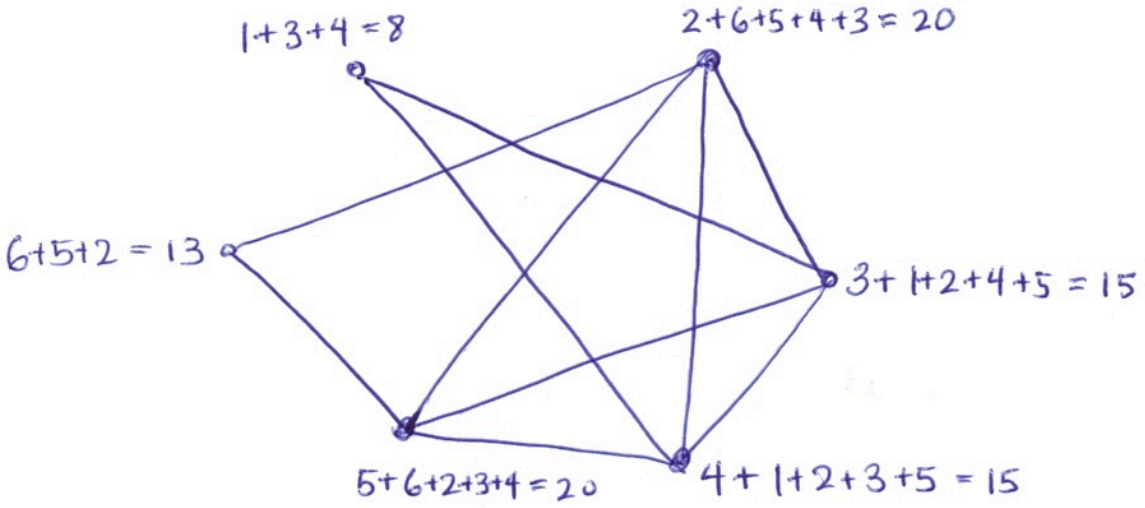
(1) Bob shares this graph, the circled vertices are his private key:



(2) Alice chooses a number: $N = 21$ She labels:



(3) Alice clumps to make new labels:



(4) Bob sums the clumped labels on the vertices in his perfect code:

$$8 + 13 = 21.$$

He got the message!

Remark: Note that the security of the system depends upon the assumption that finding a perfect code in a graph is hard. I.e., if we had an easy way of looking at Bob's graph and deducing the perfect code that he's keeping secret, then we could read all the messages being sent to Bob.

Q: How do we know it's hard to find a perfect code?

Ans: We don't know for certain.

§7.2 KidsRSA.

In KidsRSA, the "hard math problem" that will lie at the core of the security of the system will be finding inverses in modular arithmetic. Ie. solving

$$17593 \cdot x = 1 \bmod 104659$$

for x (the numbers are arbitrary).

KidsRSA cryptosystem :

Alice wants to send a message to Bob. To do this, Bob needs to make a public key and a private key, which he does as follows:

(1) Bob chooses positive integers a, b, a', b', and set $M = ab - 1$.

(2) Bob sets $e = a'M + a$, $d = b'M + b$, and $n = \frac{ed-1}{M}$.

Note that $n = \frac{(a'M+a)(b'M+b) - 1}{M}$

$$= \frac{a'b'M^2 + a'bM + ab'M + (ab - 1)}{M}$$

$$= \frac{(a'b'M + a'b + ab' + 1)M}{M}$$

So it's an integer.

The pair (e, n) is Bob's public key, the pair (d, n) is Bob's private key.

(3) Alice chooses a message to send to Bob. A message must always be an integer m with $0 < m < n$, ~~with m relatively prime to n~~. The encrypted message she sends is

$$e \cdot m \pmod{n}.$$

(4) Bob decrypts by multiplying by d and reducing mod n. This works because

$$n = \frac{ed-1}{M}, \text{ so } ed = nM+1$$

$$\Rightarrow ed \equiv 1 \bmod n.$$

So $de \cdot m \pmod{n} = m \pmod{n}$.

Of course, every eavesdropper knows Bob's public key of (e,n). So if they can compute what integer d gives $de = 1 \bmod n$, then they can read every message intended for Bob.

Example: Bob begins by generating keys. He chooses $a=5$, $b=3$, $a'=7$, $b'=5$. Then

$$M = ab-1 = 14$$

and $e = 7 \cdot 14 + 5 = 103$, $d = 5 \cdot 14 + 3 = 73$,

and $n = \frac{103 \cdot 73 - 1}{14} = 537.$

Bob publishes (103, 537) for the whole world to see.

Alice decides she wants to send the message "30" to Bob. Note $537 = 3 \cdot 179$, (179 prime), so $\gcd(30, 537) = 3$. (not a problem!)

Alice computes $30 \cdot 103 = 405 \bmod 537$ (this involves only multiplication and long division).

Bob receives the ciphertext 405. He computes

$$405 \cdot 73 \pmod{537}$$ and recovers 30.

§7.3 Breaking KidsRSA.

To decipher all messages sent using KidsRSA, we need to be quick at solving equations of the form

$$ex = 1 \bmod n$$

for "x" when we are given (e, n). One approach would be to try every $x \in \{0, \ldots, n-1\}$, which would obviously work but fails the "quick" part. There is a quick way to do it:

The Euclidean Algorithm:

This is a process for finding greatest common divisors.

We first need to make an observation.

Lemma: Suppose $0 \leq x \leq y$ are integers. Then

$$\gcd(x,y) = \gcd(x, y-x).$$

Proof: Write $d = \gcd(x,y)$ and $d' = \gcd(x, y-x)$. We'll show $d = d'$.

First, $d \mid x$ and $d \mid y$ so $d \mid (y-x)$. So d is a divisor of both x and $y-x$, meaning $d \leq d'$. On the other hand, $d' \mid x$ and $d' \mid (y-x)$ so d' divides $(y-x)+x = y$. But then d' divides both x and y, so $d' \leq d$.

Thus $d = d'$.

$=$

Here is how to use the lemma:

The gcd of two small numbers is easier to find than the gcd of two large numbers. So if $0 < x < y$, and we're asked for $\gcd(x,y)$, it'll be easier to look for $\gcd(x, y-x)$. In fact:

$$\gcd(x,y) = \gcd(x,y-x) = \gcd(x,y-x-x) = \gcd(x,y-x-x-x) = \cdots$$

- $\gcd(x,y)$: what we want
- $\gcd(x,y-x)$: easier to find
- $\gcd(x,y-x-x)$: even easier to find!
- $\gcd(x,y-x-x-x)$: even easier!

But don't subtract so many x's that $y-x-\ldots-x$ becomes negative.

Q: How many can we subtract without $y-x-\ldots-x$ being neg?

Recall that $\lfloor r \rfloor$ is the floor of r, it is the largest integer that is $\leq r$. E.g.

$\lfloor \pi \rfloor = 3$

$\lfloor \sqrt{2} \rfloor = 1$

$\lfloor 2 \rfloor = 2$.

A: We can subtract $\lfloor y/x \rfloor$ copies of x.

The Euclidean Algorithm is based on this "simplifying subtraction":

Suppose $0 \leq x < y$ are integers. This procedure finds the $\gcd(x,y)$:

(1) If $x=0$, stop. In this case $\gcd(x,y) = y$.

Otherwise proceed to Step 2.

(2) Set $x = x_0$ and $y = y_0$. Then for $i \geq 1$, set:

$$y_i = x_{i-1} \text{ and } x_i = y_{i-1} - \left\lfloor y_{i-1}/x_{i-1} \right\rfloor x_{i-1}.$$

(3) When $x_i = 0$, stop. The gcd is y_i.

Example: Suppose we want to compute the gcd of 707 and 980. Then $0 \leq 707 < 980$, and our steps are as follows:

First, $707 \neq 0$. So there's work to do.

Second, set $x_0 = 707$ $\quad y_0 = 980$.

$$x_1 = 980 - \left\lfloor \tfrac{980}{707} \right\rfloor 707 \qquad y_1 = 707.$$

$$= 980 - 707$$

$$= 273$$

$$x_2 = 707 - \left\lfloor \tfrac{707}{273} \right\rfloor 273 \qquad y_2 = 273$$

$$= 707 - 2(273)$$

$$= 161$$

$$x_3 = 273 - \left\lfloor \tfrac{273}{161} \right\rfloor 161 \qquad y_3 = 161$$

$$= 273 - 161$$

$$= 112$$

$$x_4 = 161 - \left\lfloor \tfrac{161}{112} \right\rfloor 112 \qquad y_4 = 112$$

$$= 161 - 112$$

$$= 49$$

$x_5 = 112 - \lfloor 112/49 \rfloor 49$ $\quad y_5 = 49.$

$= 112 - 2 \cdot 49$

$= 14$

$x_6 = 49 - \lfloor 49/14 \rfloor 14$ $\quad y_6 = 14.$

$= 49 - 3 \cdot 14$

$= 7$

$x_7 = 14 - \lfloor 14/7 \rfloor 7$ $\quad y_7 = 7$

$= 14 - 14 = 0$. STOP This is the gcd.

The proof that this works boils down to a long chain of equalities that use the lemma we proved. E.g. Why is $\gcd(707, 980) = 7$?

Because:

$$\gcd(707, 980) = \gcd(707, 273) = \gcd(273, 161) = \cdots$$
$$= \gcd(7, 14) = \gcd(7, 0) = 7.$$

This is great. We have a handle on gcd's, but to crack Kids RSA, we needed to solve

$$ex \equiv 1 \bmod n,$$

ie we need to compute inverses mod n (x would be called the inverse of e, written e^{-1}, in this case). For this we have the Extended Euclidean Algorithm.

The "extended" version of this algorithm is simple: Just don't forget how many multiples of x_{i-1} you subtracted from y_{i-1} when you proceed to the next line!

It allows you to do the following: If $\gcd(x,y)=d$, this algorithm will find a, b (integers) such that

$$ax+by=d.$$

Here is the extended Euclidean algorithm, done with $x=707$ and $y=980$:

$$x_0 = 707 \qquad y_0 = 980$$

$$x_1 = \underbrace{980-707}_{273} \qquad y_1 = 707.$$

$$x_2 = 707 - \overset{273}{2}(980-707) \qquad y_2 = \underbrace{980-707}_{273}.$$

$$= \underbrace{3\cdot 707 - 2\cdot 980}_{161}$$

$$x_3 = \underbrace{980-707}_{273} - \underbrace{(3\cdot 707 - 2\cdot 980)}_{161} \qquad y_3 = \underbrace{3\cdot 707 - 2\cdot 980}_{161}.$$

$$= \underbrace{3\cdot 980 - 4\cdot 707}_{112}$$

$$x_4 = (3\cdot 707 - 2\cdot 980) - (3\cdot 980 - 4\cdot 707) \qquad y_4 = \underbrace{3\cdot 980 - 4\cdot 707}_{112}.$$

$$= \underbrace{7\cdot 707 - 5\cdot 980}_{49}$$

$$x_5 = \underbrace{(3\cdot 980 - 4\cdot 707)}_{112} - 2\underbrace{(7\cdot 707 - 5\cdot 980)}_{49}$$

$$= \underbrace{13\cdot 980 - 18\cdot 707}_{14}$$

$$y_5 = \underbrace{7\cdot 707 - 5\cdot 980}_{49}$$

$$x_6 = \underbrace{(7\cdot 707 - 5\cdot 980)}_{49} - 3\cdot(13\cdot 980 - 18\cdot 707)$$

$$= \underbrace{61\cdot 707 - 44\cdot 980}_{\text{we did it!}} = 7 = \gcd(707, 980).$$

This is $ax+by=d$.

Of course, re-writing the entire Euclidean algorithm this way is not really necessary, and there are shorthand ways of keeping track of all of this information.

E.g. In the book, when they do the E.A. on 707 and 980, they write only

980
707
273
161
112
49
14
(7) ← gcd.
0

Which is our list of y_i's.

This is faster, but bad practice. It will lead to mistakes, and gives no partial points on tests/exams/assts.

Remark: It's also common to write $\gcd(x,y)=d$ as $ax+by$ by "running the Euclidean Algorithm backwards". I.e. Start with the final step:

$$7 = 49 - 3\cdot 14$$

substitute in the previous step, $112 - 2\cdot 49 = 14$:

$$7 = 49 - 3\cdot(112 - 2\cdot 49)$$

and sub in $49 = 161 - 112$

$$7 = 161 - 112 - 3\cdot(112 - 2\cdot(161 - 112))$$
$$= 7\cdot 161 - 10\cdot 112$$
$$\vdots$$
etc.

and arrive at the same as before, $7 = 61\cdot 707 - 44\cdot 980$.

How does this help crack KidsRSA? Well, suppose we want to solve

$$ex = 1 \bmod n,$$

where $\gcd(e,n)=1$, to find "d" such that (d,n) is Bob's private key.

If $\gcd(e,n)=1$, we can use the extended Euclidean algorithm to write

$$1 = ae + bn$$

But then $1 - bn = ae$

and therefore $ae \equiv 1 \bmod n$, a is the solution!
(ie, a is Bob's Key!).

Example: We already saw an example where Bob had a public key of $(103, 537)$ and Alice sent the message "30" to Bob by computing

$$30 \cdot 103 = 405 \bmod 537.$$

She sends him the message "405" via public channels.
If we want to read the message intended for Bob, we run the Euclidean algorithm on the pair $(103, 537)$.
We get:

$x_0 = 103$ $\quad y_0 = 537$

$x_1 = 537 - \lfloor \frac{537}{103} \rfloor 103$ $\quad y_1 = 103$

$= 537 - 5 \cdot 103$

$= 22$

$x_2 = 103 - \lfloor \frac{103}{22} \rfloor \cdot 22$ $\quad y_2 = 22$

$= 103 - 4 \cdot 22$

$= 15$

$x_3 = 22 - \lfloor 22/15 \rfloor \cdot 15$ $\quad y_3 = 15$

$= 22 - 15$

$= 7$

$x_4 = 15 - \lfloor 15/7 \rfloor \cdot 7$ $\quad y_4 = 7$

$= 15 - 2 \cdot 7$

$= 1$ there's the gcd.

Now go backwards:

$$1 = 15 - 2 \cdot 7$$

$$= 15 - 2 \cdot \underbrace{(22 - 15)}_{\text{sub}} = 3 \cdot 15 - 2 \cdot 22$$

$$= 3 \cdot \underbrace{(103 - 4 \cdot 22)}_{\text{sub}} - 2 \cdot 22 = 3 \cdot 103 - 14 \cdot 22$$

$$= 3 \cdot 103 - 14(537 - 5 \cdot 103)$$

$$= 73 \cdot 103 - 14 \cdot 537$$

Therefore $73 \cdot 103 = 1 + 14 \cdot 537 = 1 \bmod 537$.
So Bob's key (private key!) is 73. We calculate
$73 \cdot 403 = 30 \bmod 537$, and have discovered Alice's message to Bob!

Conclusion: KidsRSA is not secure.

Real RSA will replace the single multiplication used in encryption (for Kids RSA) with many multiplications. Instead of using a number "e" and computing "me", we will use a number "e" and compute "m^e". However, this causes a bit of a problem:

Encryption is supposed to be fast and easy. In order to "beat" modern computing power, a number "e" to be used as a key has to be pretty big (think hundreds of digits).

Then how do we compute m^e ? (mod some large n).

We cannot do:

$$m^2 = m \cdot m$$
$$m^3 = m^2 \cdot m$$
$$m^4 = m^3 \cdot m$$
$$\vdots$$
etc,

or that would require "e" multiplications. If $e \approx 10^{200}$, this would take longer than the age of the universe to do this on even the most powerful computer. So we need:

§7.4 Fast modular exponentiation.

The idea here is to do repeated squaring, and see how quickly that goes. I.e.

$$m^2 = m \cdot m$$
$$m^4 = m^2 \cdot m^2$$
$$m^8 = m^4 \cdot m^4$$
$$m^{16} = m^8 \cdot m^8$$

and in general, we can get to 2^n (exponent) in only 'n' multiplications. This raises the problem of computing m^e when e is not a power of 2.